including a service connector associated with each attached service module. The core profile engine is adapted to process access requests from applications and to return a reference to an appropriate service and the application is not required to interface with a directory service. The core profile engine as claimed in Claim 37 is not shown by the references currently of record (e.g., Chan does not show a pluggable interface attached to plug-in service modules including an initialization parameter nor a service connector adapted to process a service request and return a reference to the service module). Claim 39 (and dependent Claims 40 and 41) is provided to more particularly claim the method of providing an application with a reference to or access to a particular service as shown in Figure 3A and described at page 18 of Applicant's specification. Each step of this method is not shown or made obvious by Chan by itself or in combination with Lundin, e.g., Chan does not teach instantiating a service connector with an application and then operating the service connector to lookup an instance of a service and return a reference to that instance of the service to the application. Hence, new Claims 37-41 are believed allowable over the references currently of record in this case.

Rejections Under 35 U.S.C. § 112

Claims 1, 10, and 19 were rejected under 35 U.S.C. § 112, second paragraph, as being indefinite based on the claim phrase "gaining reference to said first said first service by said second service. Claims 1, 10, and 19 have been amended to address this indefiniteness.

Rejections Under 35 U.S.C. § 102

In the Office Action of September 3, 2002, Claims 1-5, 7, and 9-36 were rejected under 102(e) as being anticipated by U.S. Patent No. 6,360,230 ("Chan"). This rejection is traversed based on the amendments to independent Claims 1, 10, 19, and 28 and the following comments.

As stated in the background of the specification at page 5, the invention is directed to solving the problem of an application locating and accessing a needed service (often on a remotely located server computer) without having to embed all of the logic in the application that is required for finding the service and/or negotiating for a given version of that service. With varying language, independent Claims 1, 10, 19, and 28 provide devices and methods that allow an application to locate a particular service without having to include such search language, without having to utilize a directory service and well-known associated techniques, and without

having prior knowledge of the service and/or its location in a computer system.

Claim 1 is directed to a method calling for "enabling definition of a service connector interface in conjunction with said first service" (i.e., the service being accessed) and then "subsequently invoking said service connector interface in conjunction with said second service...including instantiating said service connector interface at said second service." The method then calls for "gaining reference to said first service by said second service." Claim 1 requires that a service connector interface be defined for the service being accessed but then the service connector is invoked at the service (or application) that is attempting to find and then access the service. As stated at page 19 of the specification, "the application is only initially invoking the service connector 304 and not the service 306. It is the service connector which provides the reference to the service 306 that enables the application 302 to use it." The steps of the method of Claim 1 make it unnecessary "to embed all of the logic necessary for the application 302 to determine how to get a reference directly to the service 306" within the application.

In contrast, Chan is directed to "a framework for uniform access to various different directory services" (see col. 6, lines 14 and 15). In the Chan system, "each provider of a directory service provides an implementation of these interfaces for their directory service that maps the behavior of their API set to the behavior of these OleDs interfaces. In this way, a client that is developed to use the OleDs architecture can access each of these directory services..." (see col. 6, lines 26-32) and the OleDs define a model "to assist clients in accessing the objects of a directory service" (see col. 7, lines 34 and 35). The Applicant was well aware of the use of directories and directory services as one solution to the problem of finding resources in a distributed system (see Background at page 5, lines 1-13). While providing some useful solutions, directory and meta-directory solutions "only help to catalog information that is already known and do not provide a mechanism for automatically discovering the identity and interface parameters for a network service" (page 5, lines 9-13 of Applicant's specification).

Because Chan was solving a different problem, Chan does not teach each element of Claim 1. Specifically, Chan does not teach defining a service connector interface in conjunction with said first service and then invoking the service connector interface in conjunction with the

second service including instantiating the interface at the second service. At col. 6, lines 37-47, Chan discusses "an interface for enumerating the directory services contained within the namespaces container" and that a client can discover the existence of directory services. The interface is not specific to a particular service (such as a first service) but is instead "an interface for enumerating the directory services contained within the namespaces container." After defining service connector interface for the first service, Claim 1 calls for the interface to be invoked in conjunction with the second service by instantiating at the second service. Chan at col. 7, lines 10-13 describes a binding function that invokes retrieved code but does not describe a second service (such as a client application) invoking the defined interface. Rather, Chan states that "the invoked code uses the API set of the directory service to access the identified object" without reference to an interface that is defined relative to a particular first service. Further, Chan does not teach instantiating the service connector interface at the second service. At col. 7, lines 14-25, a container object is instantiated but there is no teaching that such instantiation occurs at the second service (such as on a system running a client application in a distributed environment) or that the object is defined relative to the first service. Hence, Chan does not anticipate Claim 1. Claims 2 and 4-9 depend from Claim 1 and are believed allowable for at least the reasons for allowing Claim 1.

Independent Claim 10 is directed to a computer program product with some limitations similar to Claim 1 but further calls for gaining reference to the first service by the second service to include retrieving an instance of the first service at the service connector interface, obtaining a service reference from the first service, and then returning said reference to the second service. Hence, Claim 10 is believed allowable for the reasons for allowing Claim 1 and also because Chan at col. 7, lines 5-10 does not teach retrieving an instance of the first service at the service connector interface. Instead, Chan "uses this namespace identifier to retrieve code (e.g., in a dynamic linked library) that implements access to the identified namespace." An instance of a service (such as a first service having a defined service connector interface) is not retrieved by the Chan binding function. Chan also does not teach returning a reference to a dynamic service, such as first service, as it simply teaches returning a pointer to an object in a directory and not to a service for which an instance has been retrieved. Claims 11, 12, and 14-18 are dependent from Claim 10 and are believed allowable for at least the reasons for allowing Claim 10.

Independent Claim 19 is directed to a method similar to Claim 1 and is believed allowable for at least the reasons for allowing Claim 1 but further includes the limitation that a particular version of the first service can be specified by the second service. Chan fails to teach this added limitation. Specifically, at col. 7, lines 59-64, Chan simply refers to the idea of default namespace and directory service that allows a user to indicate what should be a default namespace for a directory service. No mention is made of services, of locating services, and of locating a particular version of a service in situations where a number of versions of a service are stored in the directory service or available on a computer system. Claim 19, and Claims 20-22 and 24-27 that depend from Claim 19, are not anticipated by Chan and are believed in condition for allowance.

Independent Claim 28 is directed to a system for providing dynamic references between services and is believed allowable for the reasons for allowing Claim 1 but also because it calls for means for enabling definition of a service connector interface in conjunction with the first service that includes means for developing a computer program module adhering to the service connector interface in conjunction with the first service. Chan does not teach developing a program module associated with a first service that adheres to a service connector interface (such as a module associated with plug-ins shown in Figure 2). At col. 6, lines 37-39 referenced by the Examiner in rejecting Claim 2, Chan discuss providing an interface but fail to provide any teaching that a module should be provided in such interface associated with each service in the directory. Hence, Claim 28 is not anticipated by Chan and Claim 28, and Claims 30-36 that depend there from, are allowable over the art of record.

Rejections Under 35 U.S.C. § 103

In the Office Action of September 3, 2002, Claims 6 and 8 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Chan further in view of U.S. Patent No. 5,339,430 ("Lundin"). The rejection of these claims based on 103(a) is respectfully traversed based on the following discussion.

Claims 6 and 8 depend from Claim 1 which is believed allowable over the art of record, and therefore, Claims 6 and 8 are believed allowable at least for the reasons for allowing Claim 1. Further, Lundin fails to overcome the shortcomings of the teachings of Chan discussed above.

Further, Claims 6 and 8 are directed to the concept of defaulting to instantiating and then referencing a latest version or latest instance of a service being accessed or requested by client application or "second service." Lundin discusses using latest or new versions of software but does not discuss defaulting to such a choice only upon a selection for a version or a particular instance not being provided by a requesting service as called for in Claims 5 and 7 from which Claims 6 and 8 depend. Hence, the combination of Chan with Lundin does not result in the method called for in Claims 6 and 8, and these claims are believed allowable over the art of record.

Conclusions

Based on the above remarks, all pending claims are believed to be allowable over the above-discussed references. Consequently, the case is believed to be in condition for allowance, and this action is respectfully requested.

A check is provided to cover the fee for additional claims, and no other fees are believed due with this response, but any fee deficiency associated with this submittal may be charged to Deposit Account No. 50-1123.

Respectfully submitted,

Date October 1, 2002

Kent A. Lembke, Reg. No. 44,866
Hogan & Hartson LLP
One Tabor Center
1200 17th Street, Suite 1500
Denver, Colorado 80202
Telephone: (720) 406-5378
Fax: (720) 406-5301

Version With Marking to Show Changes

IN THE CLAIMS:

Claims 1, 10, 19, and 28 were amended and new claims 37-41 added as follows:

1. (Amended) A method for [allowing] providing a reference to a first service to [gain reference to] a second service in a computer system comprising:

enabling definition of a service connector interface in conjunction with said first service;

subsequently invoking said service connector interface in conjunction with said second service, wherein said subsequently invoking includes instantiating said service connector interface at said second service; and

gaining reference to said first service by said second service.

10. (Amended) A computer program product comprising:

a computer usable medium having computer readable code embodied therein for [allowing] providing a reference to a first service [to gain reference] to a second service in a computer system comprising:

computer readable program code devices configured to cause said computer system to effect enabling definition of a service connector interface in conjunction with said first service;

computer readable program code devices configured to cause said computer system to effect subsequently invoking said service connector interface in conjunction with said second service; and

computer readable program code devices configured to cause said computer system to effect gaining reference to said first service by said second service, wherein said computer readable program code devices comprise:

computer readable program code devices configured to cause said computer system to effect retrieving a service instance at said service connector interface;

computer readable program code devices configured to cause said computer system to effect obtaining a service reference from said first service; and

computer readable program code devices configured to cause said computer system to effect returning said service reference obtained from said first service to said

second service.

19. (Amended) A method for [allowing] providing a reference to a first service [to gain reference] to a second service in a computer system comprising:

providing for enabling definition of a service connector interface in conjunction with said first service;

providing for specifying a particular version of said first service that said second service desires to invoke;

providing for subsequently invoking said service connector interface in conjunction with said second service; and

providing for gaining reference to said first service by said second service.

28. (Amended) A system for providing dynamic references between services in a computer system comprising:

means for enabling definition of a service connector interface in conjunction with said first service, said definition enabling means includes means for developing a computer program module adhering to said service connector interface in conjunction with said first service;

means for subsequently invoking said service connector interface in conjunction with said second service; and

means for gaining reference to said first service by said second service.

37. (New) A core profile engine for use in gateway or firewall servers for enabling client applications to access plug-in service modules in a distributed computing environment without embedding location and negotiation logic within the client applications, the engine comprising:

an application programming interface in communication with the client applications and adapted with interfaces for processing a request for a service provided by one of the service modules; and

a pluggable interface attaching to the plug-in service modules, wherein the attaching includes providing an initialization parameter comprising a storage location for each of the plug-in service modules;

wherein the pluggable interface further includes a service connector associated with each of the attached plug-in service modules that is adapted to receive the service request from the application programming interface and to return a reference to the one service module providing the service based on the storage location.

38. (New) The engine of claim 37, wherein the plug-in service modules are selected from a group consisting of an authorization plug-in, an authentication plug-in, a notification plug-in, a log plug-in, a group plug-in, an entity identification factory plug-in, and a replication plug-in.

39. (New) A computer-implemented method for providing an application with access to a service without the use of a directory service, comprising:

reading a configuration file with the application to determine an indicator to the service;

instantiating with the application a service connector for the service based on the indicator;

requesting with the application that the service connector provide access to the service;

first operating the service connector to lookup an instance of the service;

second operating the service connector to obtain a reference to the instance of the service; and

third operating the service connector to return the reference to the instance of the service to the application.

40. (New) The method of claim 39, wherein the requesting includes an identification of a version of the service for which the application is requesting access.

41. (New) The method of claim 39, further including operating the application to request identification of an interface implemented by the referenced service and operating the service connector to retrieve and return the interface identification to the application for use in utilizing the referenced service.